PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-219810
(43)Date of publication of application : 18.08.1995

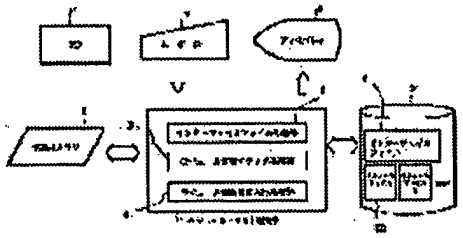
(51)Int.Cl. G06F 11/28
G06F 9/06

(21)Application number : 06-012701
(22)Date of filing : 04.02.1994

(71)Applicant : CANON INC
(72)Inventor : OTA SATOSHI

(54) MODULE TESTING METHOD AND DEVICE THEREFOR

(57)Abstract:
PURPOSE: To improve efficiency of the single program module test.
CONSTITUTION: An interface file processing part 2 reads a file that includes the function to be tested and procudes the name of this function and a list of input/output parameters. A module definition editor processing part 3 inputs the name of the function and the parameter name of the function based on the parameter list. A module function automatic execution processing part 4 displays the input parameter name, etc., and inputs the proper value. Then the value is outputted in accordance with the set parameter value when the execution of the function is instructed. Therefore it is not required to produce the test programs for each module to be tested, and the test efficiency is improved.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平7－219810

(43)公開日　平成７年(1995)８月18日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/28	３４０　Ａ	9290－5B		
9/06	５４０　Ｕ	9367－5B		

審査請求　未請求　請求項の数６　ＯＬ　（全 12 頁）

(21)出願番号　特願平6－12701

(22)出願日　平成６年(1994)２月４日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　太田　聡
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

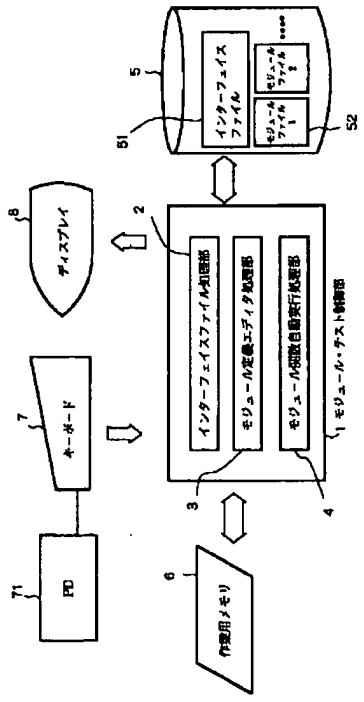
(74)代理人　弁理士　大塚　康徳　(外１名)

(54)【発明の名称】　モジュールテスト方法および装置

(57)【要約】

【目的】プログラムモジュール単体テストの効率を向上させる。

【構成】インターフェイスファイル処理部２は、テスト対象の関数が存在するファイルを読み込んで、テスト対象の関数名や入出力パラメータのリストを作成する。モジュール定義エディタ処理部３は、前記リストに従ってテスト対象の関数名や、その関数のパラメータ名を入力させる。モジュール関数自動実行処理部４は、入力されたパラメータ名等を表示し、適当な値を入力させる。その後、関数の実行が指示されると、設定されたパラメータの値に応じた値を出力する。このようにして、テスト対象モジュール毎にテストプログラムを作成する必要がなくなり、テストの効率を向上できる。



【特許請求の範囲】
【請求項１】　ディスプレイを有する情報処理システムにおけるプログラムモジュールテスト方法であって、
テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み工程と、
前記読み込み工程により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示工程と、
該表示工程により表示される画面に応じてデータを入力する入力工程と、
該入力工程により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する工程と、を備えることを特徴とするモジュールテスト方法。
【請求項２】　前記表示工程は、設定させるデータをメニュー形式で表示し、前記入力工程は前記メニューに対応したデータを入力することを特徴とする請求項１記載のモジュールテスト方法。
【請求項３】　前記インターフェイス情報は、前記テスト対象プログラムの名称と該プログラムの入出力パラメータとを含むことを特徴とする請求項１または請求項２記載のモジュールテスト方法。
【請求項４】　プログラムをモジュール毎にテストするモジュールテスト装置であって、
テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み手段と、
前記読み込み手段により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示手段と、
該表示手段により表示される画面に応じてデータを入力する入力手段と、
該入力手段により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する手段と、を備えることを特徴とするモジュールテスト装置。
【請求項５】　前記表示手段は、設定させるデータをメニュー形式で表示し、前記入力手段は前記メニューに対応したデータを入力することを特徴とする請求項４記載のモジュールテスト装置。
【請求項６】　前記インターフェイス情報は、前記テスト対象プログラムの名称と該プログラムの入出力パラメータとを含むことを特徴とする請求項４または５記載のモジュールテスト装置。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、例えばサンプル・プログラム等を作成することなく、作成されたモジュールの動作の確認ができるモジュールテスト方法およびそれを用いた装置に関するものである。
【０００２】
【従来の技術】従来、例えばアプリケーションプログラム等、最終的な目的となるプログラム（以下、目的プログラムと呼ぶ）を作成する際には、それを複数の機能モジュールに分割して機能モジュール毎に作成し、それらをリンクするという手法がしばしば行われる。この場合、目的プログラムの動作は各機能モジュール毎に単体テストを行って検証されるのが普通である。この各機能モジュール毎の単体テストに於ては、作成者自身が目的プログラムとは別個に簡単なサンプル・プログラムを作成し、このサンプル・プログラムにテストの対象となる各機能モジュールをリンクさせ、リンク後のサンプル・プログラムを実行して機能モジュールの動作確認を行っていた。
【０００３】
【発明が解決しようとする課題】上記従来例では、目的とするプログラムとは別に“サンプル・プログラムの作成”という本来の目的とは無関係な作業が発生しており、また、目的プログラム毎にサンプル・プログラムを作成、変更しなければならず、開発効率の低下の原因となっていた。
【０００４】本発明は上記従来例に鑑みてなされたもので、統一された、再利用可能なモジュールテスト方法およびそれを用いた装置を提供することにある。
【０００５】
【課題を解決するための手段】上記目的を達成するために本発明のモジュールテスト方法は次のような構成から成る。すなわち、ディスプレイを有する情報処理システムにおけるプログラムモジュールテスト方法であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み工程と、前記読み込み工程により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示工程と、該表示工程により表示される画面に応じてデータを入力する入力工程と、該入力工程により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する工程とを備える。
【０００６】また、本発明のモジュールテスト装置は次のような構成から成る。すなわち、プログラムをモジュール毎にテストするモジュールテスト装置であって、テスト対象プログラムに記載されたインターフェイス情報を読み込む読み込み手段と、前記読み込み手段により読み込まれたインターフェイス情報に基づいてデータを設定させる画面を前記ディスプレイに表示する表示手段と、該表示手段により表示される画面に応じてデータを入力する入力手段と、該入力手段により入力されたデータと前記インターフェイス情報とに基づいて前記テスト対象プログラムを実行する手段とを備える。
【０００７】
【作用】上記構成により、テスト対象プログラムのインターフェイス情報に基づいてデータを設定させる画面を

表示し、その画面に応じたデータを入力させ、入力されたデータに基づいてテスト対象プログラムを実行する。

【０００８】

【実施例】以下、本発明の実施例を図面を参照して詳細に説明する。

【０００９】図１は、本実施例である、Ｃ言語で記述されたモジュールの機能を検証するテストシステムの構成示す。Ｃ言語においては手続きは関数という形式で表される。目的プログラムは、１または複数の関数を記述したモジュールファイルとして作成されたプログラムをリンクして得られるが、本テストシステムはこのモジュールファイル毎に、更にそのなかの関数毎に単体テストを行うものである。

【００１０】まず、Ｃ言語で記述されたプログラムの構造を簡単に説明しておく。

【００１１】モジュールファイル内で定義された関数には関数名が付加されている。また、その関数自体の定義の前に、その関数から返される値のデータ型が、入力パラメータとともに宣言されているのが普通である。この宣言をプロトタイプ宣言と呼ぶ。プロトタイプ宣言は関数を後方参照で呼び出す場合に、その呼び出しに先立って必要とされるもので、前方参照で呼び出す場合にはプロトタイプ宣言が省かれることもある。関数を呼び出す際には、関数が返す値を受ける出力変数と関数名と入力パラメータの並びとを、通常の関数と同様の形式で記述すればよい。また、複数の関数にわたって共通に使用される変数をグローバル変数とよび、その旨定義を記述する必要がある。

【００１２】図１に於て、１は本実施例のモジュール単体テスト制御部である。モジュール単体テスト制御部１は、指定されたモジュール・ファイルから、テスト対象とする関数（テスト関数と呼ぶ）のプロトタイプ宣言部分を読み込み、インターフェイス・リストを作成するインターフェイス・ファイル処理部２と、図２に示すような対話型の画面フォームの各入力・列挙フィールドにインターフェイス・リストの内容を表示し、参照されるグローバル変数等の定義・編集を行いインターフェイス・ファイルを作成するモジュール定義エディタ処理部３と、インターフェイス・ファイルに記載された各テスト関数の項目を読み込み、図３に示すような対話形式の画面を使って、入力フィールドに代入された値をもとに、テスト関数を逐次実行するモジュール関数自動実行処理部４とを有する。５は磁気ディスク等の外部記憶装置であり、そこにはインターフェイス・ファイル５１や、モジュール・ファイル５２が格納されている。６はモジュールテスト制御部が使用する作業用メモリ、７はキーボード、７１は表示画面に対応した座標を指定するポインティングデバイス、８はディスプレイである。なお、モジュールテスト制御部１は、ＣＰＵによりプログラムを実行することで実現できる。この場合、このＣＰＵはテストシステム全体を制御するものであってもよい。

【００１３】図２はモジュール定義エディタ３による表示画面の例を示す。

【００１４】オペレータはこの画面によりテスト関数を選び、インターフェイス・ファイル処理部２にインターフェイス・リストを作成させ、それを元にして各フィールドを埋め、モジュール定義エディタ処理部３にインターフェイス・ファイル５１を作成させる。なお、図に於いて、各枠は候補を表示する為のフィールドであり、その右横の下向き矢印はプルダウン表示を行うためのスイッチである。この矢印をオペレータが指定することで、現在フィールドに表示されていない他の候補を表示することができる。

【００１５】図２に於て、９はモジュール・ファイル選択フィールドを示し、オペレータはテストの対象となるモジュールのソースファイル名を選択する。１０はテスト関数名選択フィールドを示し、テストの対象となるテスト関数名を選択する。１１は変数名入力フィールドを示し、ソースファイルを読み込んだ後、それに記述された変数名を表示するが、変更も可能である。１２は変数型タイプ選択フィールドを示し、インクルード・ファイルに宣言された変数型のタイプを列挙し、それから選択する。インクルードファイルとは、モジュールファイルにその一部として組み込まれる独立したファイルであり、各モジュールに共通して用いられるもの、例えば変数型の宣言等に用いられる。１３はパラメータ番号表示フィールドを示し、引数として渡されるパラメータの順番を示す。１４はパラメータ追加フィールドを示し、５つ以上のパラメータを持つテスト関数に於て、画面に直接表示できないパラメータの設定を行う為のものである。１５はグローバル変数入力フィールドを示し、テスト関数内部で参照されるグローバル変数の名前を設定する。１６はモジュール・ファイル読込みボタンを示し、オペレータがこれを選択すると、インターフェース・ファイル処理部２は選択されたモジュール・ファイルを読み込み、インターフェイス・リストを作成する。１７はインターフェイス・ファイル保存ボタンを示し、全ての設定をインターフェイス・ファイル５に保存する。

【００１６】このようにして作成されたインターフェイス・ファイルは、テスト関数を実行するのに必要なパラメータや関数等のアドレスに関する情報を含んでおり、それに基づいて、モジュール関数自動実行処理部４は、指定されたテスト関数を実行する。モジュール定義エディタ処理部３とモジュール関数自動実行処理部４とは、連続してその処理を行うようにしてあってもよいし、夫々の処理を独立させ、オペレータがそれぞれの実行を指定することで独立して実行するようにしてあってもよい。

【００１７】図３は、モジュール関数自動実行処理部により関数を実行する際の表示画面の一例を示す。

【００１８】図３に於て、１８はインターフェイス・ファイル選択フィールドを示し、オペレータはテストしようとする関数にかかわるインターフェイス・ファイルを選択する。１９は関数名選択フィールドを示し、テストしようとする関数を選択する。２０はパラメータ値入力フィールドを示し、パラメータ変数値、あるいは即値を入力する。パラメータ値入力フィールド２０は、４つまでのパラメータについてはその名を羅列する形式で表示され、オペレータは羅列された名前の変数にたいする値を入力する。２１はパラメータ選択フィールドを示し、５つ目以上のパラメータについての入力を施す。２２は返却値表示フィールドを示し、返却値の値を表示する。２３は返却値の内容表示フィールドを示し、返却値が参照する領域の内容を表示する。２４はグローバル変数選択フィールドを示し、テスト関数で参照されるグローバル変数を選択する。２５は変数値表示フィールドを示し、選択されたグローバル変数の値を表示する。２６はテスト関数を実行する実行ボタン、２７は終了ボタンを示す。

【００１９】上記各フィールドの内容としては、変数やパラメータおよびボタン類以外のものについては、モジュール定義エディタ処理部３により作成されたインターフェイス・ファイル５１に登録されている値が候補として表示され、オペレータはそのうちから所望のものを選ぶことになる。

【００２０】以上のようにして、オペレータはテストしようとする関数および関数のパラメータ等、実行に必要な情報を対話形式で指定することができる。このため、テスト関数を呼び出すテストプログラムをテスト関数毎に作成する手間を省くことができる。

【００２１】以上、オペレータとテストシステムとのインターフェイスについて説明した。次に、モジュールテスト制御部１における各処理の手順を説明する。

【００２２】図４はインターフェイス・ファイル作成処理部の処理手順を示すフローチャートである。

【００２３】本実施例においては、テストの対象となるモジュールの実行ファイル（オブジェクト，ライブラリ等）がリンクされたときに、そこで作成されるマップ・ファイルを必要とする。マップファイルとは、リンクされたファイルの状態を記録したファイルであり、リンクされたファイルの名前やそのアドレス、リンク対象となる関数の名前やアドレス等が記録されている。

【００２４】まず、ステップ４Ｓ１にて、予め、既定の初期状態にあるインターフェイス・ファイル（以下、DEFAULT.ITF と呼ぶ）に準備された変数用の#typedef型宣言を、図１の作業用メモリ６に読み込む。以降、ファイルの読み込み、インターフェイス・ファイルの編集等は作業用メモリ６内で行われる。DEFAULT.ITF は、図５に示すようなセクション項目を持ち、項目５－１には、Ｃ言語開発用に提供されているインクルード・ファイルに定義されているものが含まれ、各定義語にはスタック参照の引渡しタイプ（VALue:即値，OFFset: オフセット値，SEGment ： セグメント値）とサイズが指定されている。

【００２５】ステップ４Ｓ２にて、指定されたモジュール・ファイルとリンク時に作成されたマップ・ファイルを読み込む。ファイルが指定されていない場合、カレント・ディレクトリにある全てのモジュール・ファイルとマップ・ファイルを読み込む。

【００２６】ステップ４Ｓ３にて、モジュール・ファイルが参照している独自のインクルード・ファイルを読み込む。

【００２７】ステップ４Ｓ４にて、テスト対象のモジュール・ファイル名を、図５の５－２に示す［＄＄Module_Section］項目の項目名として登録する。

【００２８】ステップ４Ｓ５にて、インクルード・ファイルで定義された変数型を示す#typedef宣言文を取り出し、ステップ４Ｓ６にて、取り出した宣言文を、図５の項目５－３に示す、［＄＄Include_Section ］項目に追加する。

【００２９】ステップ４Ｓ７にて、プロトタイプ宣言文を判別するキーワードがあるか否かを判断する。このキーワードは、予め、モジュール・ファイルに明示的に記述されているものであるが、無い可能性もあるので、そのための処理としてステップ４Ｓ１０以降の処理がある。判断の結果、キーワードがあれば、ステップ４Ｓ８にて、それに続くプロトタイプ宣言文を取り出す。

【００３０】ステップ４Ｓ９にて、取り出したプロトタイプ宣言文から、図５の項目５－４の［＄＄Interface_Section ］に示すような、インターフェイス・リスト構造体を作成する。このとき、マップ・ファイルにある同名のテスト関数を検索し、そのテスト関数のアドレスを構造体に追加する。１つのインターフェイス・リストには次のリストを示すポインタが含まれ、リレーショナルなリストが作成される。この処理をキーワードを発見できる間繰り返す。

【００３１】キーワードが発見できなくなったならば、ステップ４Ｓ１０にて、キーワード無しのプロトタイプ宣言文を取り出すために、読み込んだファイルの読み込みボタンを先頭に戻す。ステップ４Ｓ１１にて、ファイルの終了かどうかを判断する。判断の結果、終了でなければ、ステップ４Ｓ１２にて、テスト関数の中でキーワードの無いプロトタイプ宣言文を検索する。ステップ４Ｓ１３にて、テスト関数が見つかったかどうかを判断し、見つかったならば、ステップ４Ｓ１４にて、プロトタイプ宣言文を取り出す。ステップ４Ｓ１５にて、取り出したプロトタイプ宣言文から、図５の項目５－４の［＄＄Interface_Section ］に示すような、インターフェイス・リスト構造体を作成する。

【００３２】判断の結果、見つからなかったならば、ス

テップ４Ｓ１６にて、マップ・ファイルに記述されているグローバル変数名とアドレスを、図５の項目５－５に示す［＄＄Global_Section］に追加し、インターフェイス・ファイル処理部２における処理を終了する。

【００３３】以上のようにして、インターフェイス・ファイル５１には、テスト対象となるモジュール・ファイルから外部につながる関数や変数の名前及びそれらのアドレスが登録される。

【００３４】図６はモジュール定義エディタ処理部３の処理手順を示すフローチャートである。これは、図２の画面に応じてユーザが操作する手順に従ったイベントの発行によって、処理が振り分けられる、イベント駆動型の処理となる。

【００３５】図２のモジュール選択フィールド９の右端のスクロールボタンが押下されたならば、ステップ６Ｓ１にて、テスト対象となるモジュール・ファイル名がプルダウン形式で列挙される。ステップ６Ｓ２にて、列挙されたファイルの中から選択されたファイル名を１つ取得し、モジュール・ファイル変数として作業メモリ６に格納する。各フィールド毎に割り当てられた変数データは、保存段階で、前述のインターフェイス・ファイル作成処理部に於て作成されたインターフェイス・ファイルの各セクションの新規項目として登録される。

【００３６】モジュール・ファイル読み込みボタン１６が押下されると、ステップ６Ｓ１４にて、選択されたモジュール・ファイルを作業メモリ６に読み込む。この時、既にインターフェイス・ファイルが存在するならば、そのインターフェイス・ファイルを読み込み、存在しなければ、ステップ６Ｓ１５にて、読み込んだファイルから、前述のインターフェイス・ファイル作成処理部２を使って、インターフェイス・リストを作成する。ステップ６Ｓ１６にて、インターフェイス・リストの最初のテスト関数の内容を各フィールドに割り当て、表示する。この一連の処理は、複数のモジュール・ファイルが存在するときの選択手段であり、１つしか存在しない場合は、モジュール・ファイル読み込みが自動的に実行され、最初のテスト関数についての各項目が表示される。

【００３７】以下は、インターフェイス・リストの項目を変更，追加、あるいは、修正する場合の処理についてである。

【００３８】関数名選択フィールド１０では、右端のスクロールボタンが押下されたならば、ステップ６Ｓ３にて、テスト対象となる関数名がプルダウン形式で列挙される。ステップ６Ｓ４にて、列挙された関数の中から、選択された関数名を取得し、関数名変数として、作業メモリ６に格納する。このとき、前回とは違う別の関数が選択されたならば、前記関数についての各フィールド値をインターフェイス・リスト形式に変換して、作業メモリ６内に保存する。ステップ６Ｓ５にて、関数名をもとにして、作業メモリ６上のインターフェイス・リストから関数名の合致したセクション項目を取得する。ステップ６Ｓ６にて、取得したセクションの［＄＄Interface_Section］に記述された項目を各フィールドの項目列挙変数に代入し、表示する。

【００３９】変数名入力フィールド１１では、ステップ６Ｓ７にて、ユーザが入力，変更した変数名を取得する。ここで入力された変数名は、インターフェイス・リストのパラメータ名に反映される。

【００４０】変数型タイプ選択フィールド１２では、右端のスクロールボタンが押下されたならば、ステップ６Ｓ８にて、インターフェイス・ファイル５１の項目５－１，５－３に示される#typedef宣言文がプルダウン形式で列挙される。ステップ６Ｓ９にて、選択された変数型タイプを取得し、パラメータと対にして、型タイプ変数あるいは返却値変数として作業メモリ６に保存する。

【００４１】パラメータ番号表示フィールド１３では、ステップ６Ｓ１０にて、パラメータの引渡し順序を昇順で表示する。１～４つ目までのパラメータは、一度に表示できるが、５つ目からは、パラメータ追加フィールド１４を使って、ステップ６Ｓ１１にて、選択されたパラメータ順に、パラメータの設定を列挙する。ステップ６Ｓ１２にて、選択された引渡し番号を取得し、上記の変数名入力フィールド、変数型タイプ選択フィールドと同様の処理を実行し、５つ目以降のパラメータの設定を行う。

【００４２】グローバル変数入力フィールド１５では、ステップ６Ｓ１３にて、テスト関数で参照されるグローバル変数名を入力する。ステップ６Ｓ９にて、選択されたグローバル変数型タイプを取得し、変数名と変数型タイプのペアを作業メモリ６に保存する。複数のグローバル変数が参照される場合、名前とタイプが決定されたならば、右端のスクロール・ボタンを使って次の入力フィールドを表示し、新規入力する。これを、参照されるグローバル変数全てについて行う。ここ迄の処理をモジュール・ファイル内のテスト対象となる関数全てについて同様の操作手順で行う。

【００４３】次に、インターフェイス・ファイル保存ボタン１７では、ステップ６Ｓ１６にて、作業メモリ６に保存された各テスト関数のデータをインターフェイス・リスト形式に変換して、モジュール・ファイル名の拡張子を、".ITF"に変更した名前で、インターフェイス・ファイルとして図示した外部記憶装置５に保存する。

【００４４】以上の手順により、テスト関数を含むモジュールファイルのインターフェイス・ファイルを作成する。

【００４５】図７はモジュール関数自動実行処理部の処理手順を示すフローチャートである。これは、ユーザの操作手順に従ったイベントの発行によって、処理が振り分けられる、イベント駆動型の処理となる。

【００４６】図３のインターフェイス・ファイル選択フ

ィールド１８では、右端のスクロール・ボタンが押下されたならば、ステップ７Ｓ１によって、上記モジュール定義エディタによって作成されたインターフェイス・ファイルをプルダウン形式で列挙する。ステップ７Ｓ２にて、ユーザによって選択されたインターフェイス・ファイル名を取得する。ステップ７Ｓ３にて、インターフェイス・ファイルを作業メモリ６に読み込み、［$$Interface_Section］に登録されたテスト関数の名前を、関数名選択フィールド１９の列挙項目とする。

【００４７】右端のスクロール・ボタンが押下されたならば、ステップ７Ｓ４にて、ステップ７Ｓ３で設定されたテスト関数名をプルダウン形式で列挙する。ステップ７Ｓ５にて、列挙された関数名から選択されたテスト関数名を取得する。ステップ７Ｓ６にて、テスト関数に応じたインターフェイス・リストから［＄＄Interface_Section］項目を取り出し、その中の１～４つ迄の入力パラメータ名を表示し、５つ目以降のパラメータ名をパラメータ選択フィールド２１の列挙項目に加える。

【００４８】ステップ７Ｓ７にて、パラメータの値を入力値フィールド２０に入力する。入力された値は、インターフェイス・リストのパラメータ型タイプによって、アドレス参照か、即値か決定される。パラメータ選択フィールド２１とそれに続くパラメータ値入力フィールド２０には、ステップ７Ｓ８にて、５つ目以降のパラメータ名を列挙し、ステップ７Ｓ９にて、パラメータ値を順次入力する。この結果、５つ目以降のパラメータについての入力操作ができる。

【００４９】グローバル変数選択フィールド２４では、ステップ７Ｓ１２にて、インターフェイス・ファイルの項目５－５に登録されたグローバル変数名が列挙される。変数値表示フィールド２５では、ステップ７Ｓ１３にて選択された変数名を取得し、ステップ７Ｓ１４にて変更される前の値を設定する。このフィールドは関数実行前は値の入力を受け付け、実行後変更された変数の値を表示する。ここまでの処理でテスト関数のパラメータ入力が終了する。

【００５０】次に、実行ボタン２６が押下され、ステップ７Ｓ１６にて、テスト関数が実行される。詳細は、図８に示す。

【００５１】ステップ７Ｓ１６－１にて、設定されたパラメータ値及びグローバル変数値を取得する。ステップ７Ｓ１６－２にて、パラメータ変数用の作業領域を作業メモリ６に確保する。ステップ７Ｓ１６－３にて、テスト関数名からインターフェイス・ファイルに記述されたセクション項目の内容を作業メモリ６に取り出す。このセクション項目のパラメータ・タイプ、関数アドレスに従って、テスト関数が実行される。

【００５２】ステップ７Ｓ１６－４にて、テスト関数内で参照されるグローバル変数があるかを判断する。判断の結果、参照があれば、ステップ７Ｓ１６－５にて、上記処理で入力された変数値を設定する。

【００５３】ステップ７Ｓ１６－６にて、パラメータの引渡しが終了したかを判断する。判断の結果、終了でなければ、ステップ７Ｓ１６－７にて、読み込んだインターフェイス・ファイルから、テスト関数に引き渡される全てのパラメータ・タイプを取得する。ステップ７Ｓ１６－８にて、パラメータがセグメント間の参照かを判断する。判断の結果、セグメント参照であるならば、ステップ７Ｓ１６－９にて、作業メモリ６に割り当てられたＤＡＴＡセグメントの値をスタックにＰＵＳＨする。セグメント参照でなければ、ステップ７Ｓ１６－１０にて、アドレス参照かを判断する。判断の結果、アドレス参照ならば、ステップ７Ｓ１６－１１にて、パラメータのオフセット値をスタックにＰＵＳＨする。アドレス参照でなければ、ステップ７Ｓ１６－１２にて、パラメータの即値をスタックにＰＵＳＨする。

【００５４】以上の処理を、パラメータの引渡しが続く間行い、ステップ７Ｓ１６－６の判断によって、引渡しが終了したならば、ステップ７Ｓ１６－１３にて、インターフェイス・ファイルの関数名に対応したアドレス値から、テスト関数のＣＯＤＥセグメントと、テスト関数の呼び出しアドレスを取得し、ステップ７Ｓ１６－１４にて、テスト関数を実行する。処理の結果は、ステップ７Ｓ１６－１５にて、返却値、グローバル変数の表示用のフィールド項目に設定する。

【００５５】ステップ７Ｓ１６の詳細は以上のようなものである。

【００５６】ステップ７Ｓ１０にては、返却値の値を表示する。ステップ７Ｓ１１にて、返却値がアドレス参照値であれば、参照される領域の内容を表示する。また、グローバル変数の変更結果については、入力手順と同様に、ステップ７Ｓ１２，１３にて、グローバル変数を選択し、ステップ７Ｓ１５にて、選択されたグローバル変数の値を表示する。以上の処理によって、テスト対象となる関数単位でのテストが可能になる。

【００５７】このように、本実施例のテストシステムでは、モジュール・ファイルに対応したインターフェイス・ファイルを準備することで、いかなるモジュールに対しても統一的な操作環境を提供し、モジュール関数単体の実行テストを可能にする。オペレータはモジュールの内容にかかわらず統一された形式の手順で、統一された画面表示を見ながらテスト対象となるモジュールの検証を行うことができる。そのため、モジュールごとにテストプログラムの作成といった作業が不要となり、効率的にモジュールテストを行うことができる。

【００５８】また、オペレータはメニュー形式の画面を参照しつつ対話形式でテストを遂行することができるため、パラメータの値変更やテスト対象の関数の変更が容易にでき、効率的にテストを行うことができる。

【００５９】また、本テストシステムは、プログラムの

開発環境と同一のコンピュータシステム内に備えることができる。そのため、モジュール開発後時間をおかずにそのテストを行うことができる。

【００６０】

【他の実施例】本実施例では、モジュール関数自動実行処理部４の中に関数呼び出しの機構を設けているが、この機構をコマンド・インタプリタに切り替えることで、自作したコマンド・ユーティリティ、あるいは、ソース・ファイルの無い、市販のパラメータ入力を持ったコマンド・ユーティリィに対して単体テストの環境を提供することができる。

【００６１】また、本実施例はＣ言語で作成されたプログラムに関して説明したが、他の言語で記述された場合であってもサブルーチン単位でモジュール化されることは通常行われることであり、そのような場合にも各モジュールファイル間をつなぐための変数やサブルーチンなどの名前やアドレスに関する情報、すなわちテスト態様となるプログラムモジュールとそれを用いるプログラムとのインターフェース情報を用い、パラメータとして適当な値をセットさせ、テスト対象プログラムを呼び出せば、本実施例と同様にテストシステムを構成することができることは容易に理解できる。

【００６２】尚、本発明は、複数の機器から構成されるシステムに適用しても、１つの機器から成る装置に適用しても良い。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。

【００６３】

【発明の効果】以上説明したように、本発明に係るモジュールテスト方法およびそれを用いた装置とシステムによれば、プログラムの開発において統一的なモジュール単体テストが可能になり、開発効率を向上させるという効果を奏する。

【００６４】

【図面の簡単な説明】

【図１】本発明の実施例を示すテストシステムのブロック図である。

【図２】ダイアログ型モジュール定義エディタの表示画面の概要図である。

【図３】モジュール関数自動実行エディタによる表示画面の概要図である。

【図４】インターフェイス・ファイル作成処理部のフローチャートである。

【図５】インターフェイス・ファイルの書式である。

【図６】モジュール定義エディタ処理部のフローチャートである。

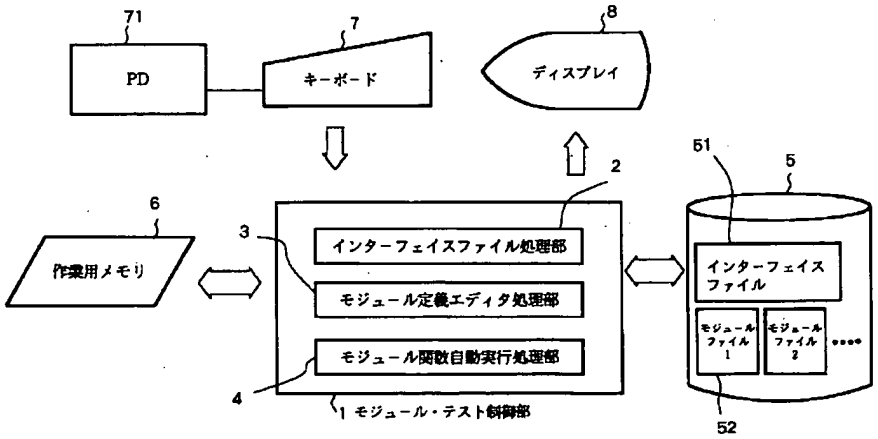
【図７】モジュール関数自動実行処理部のフローチャートである。

【図８】モジュール関数自動実行処理部のフローチャートである。

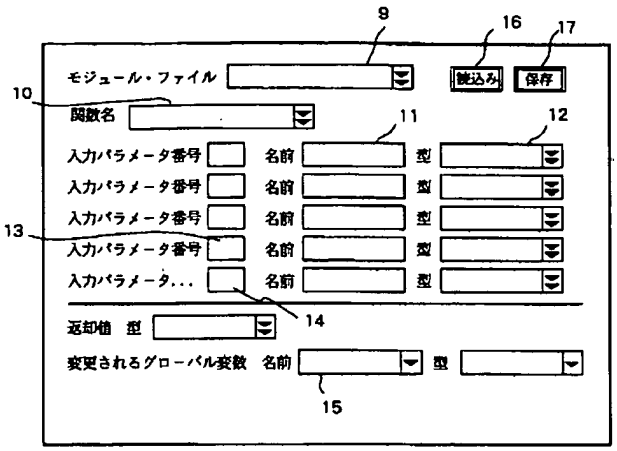
【符号の説明】

１　モジュール単体テスト制御部
２　インターフェイス・ファイル作成処理部
３　ダイアログ型モジュール定義エディタ処理部
４　モジュール関数自動実行処理部
５　インターフェイス・ファイル（外部記憶装置）
６　作業用メモリ
７　キーボード
８　ディスプレイ
５２　モジュールファイル（外部記憶装置）
７１　ポインティング・デバイス

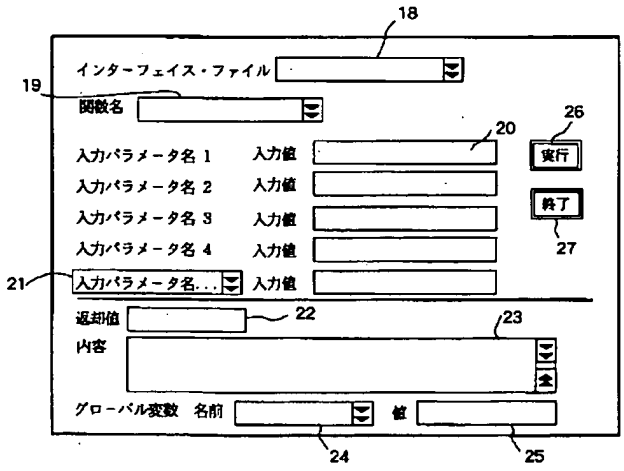
【図１】



【図2】

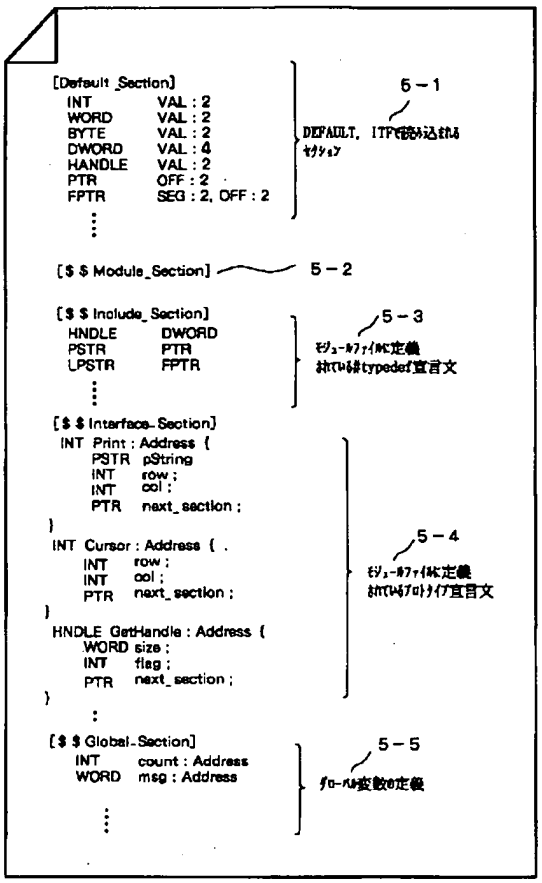


【図3】

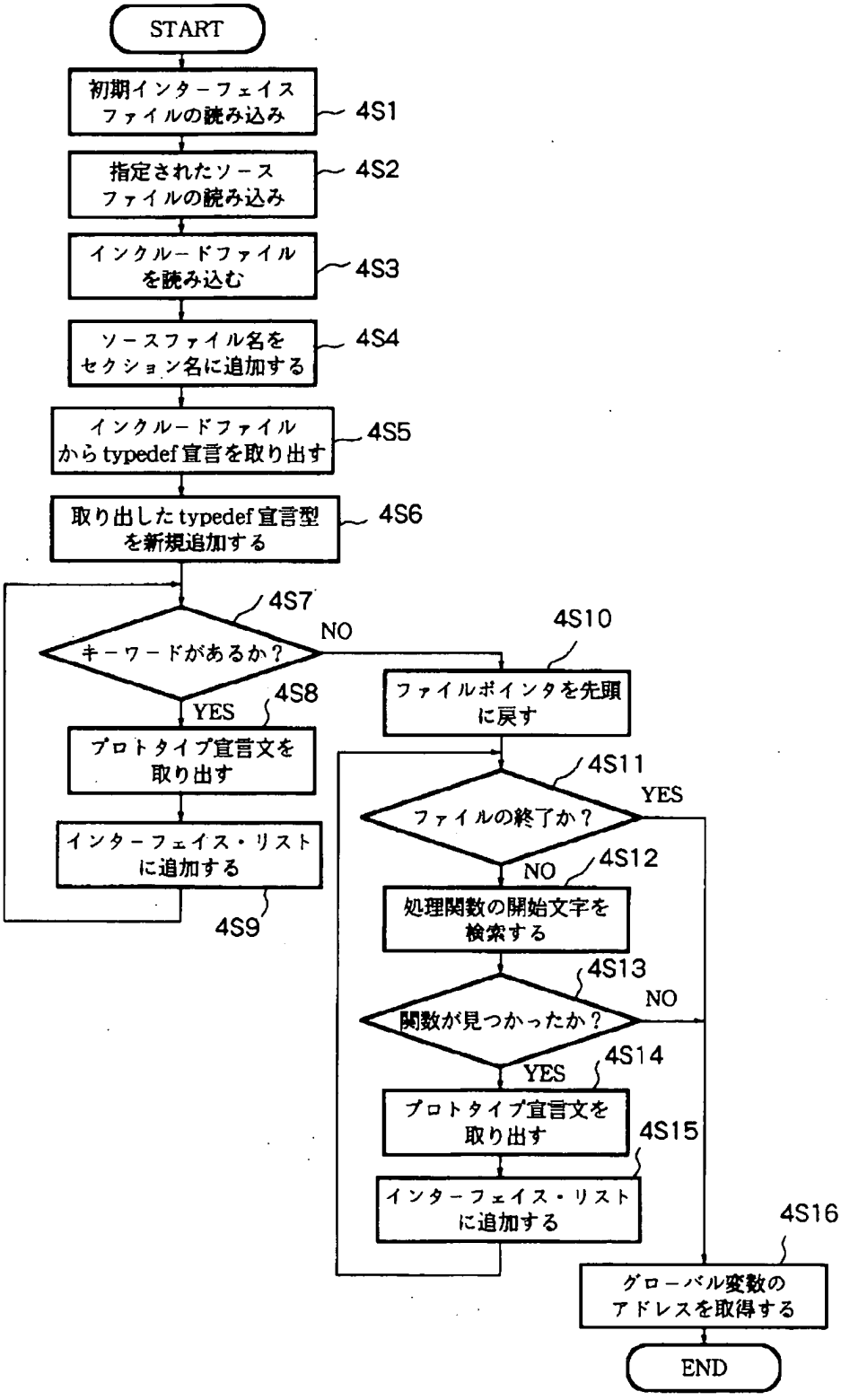


【図5】

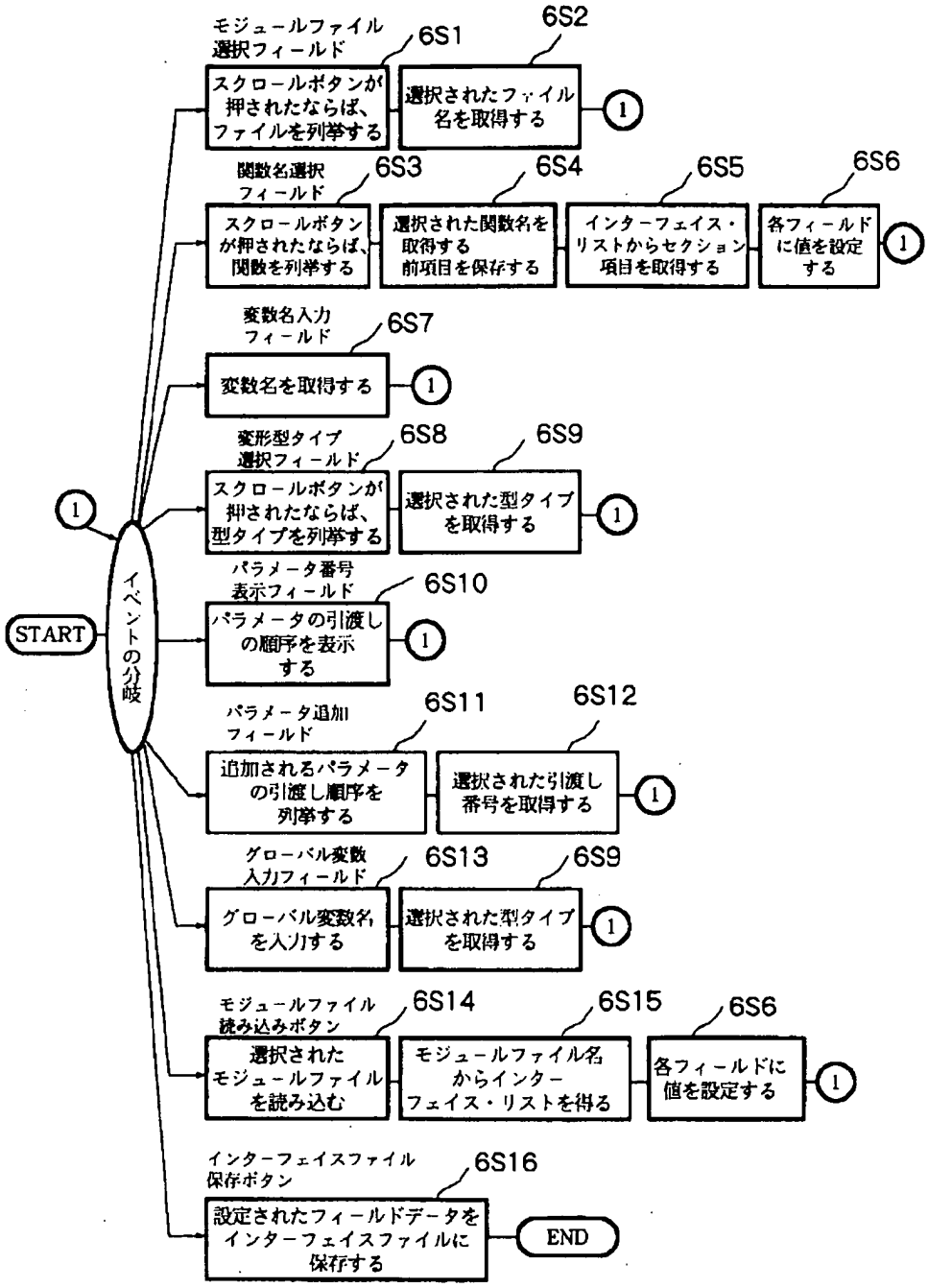
インターフェイス・ファイル書式図



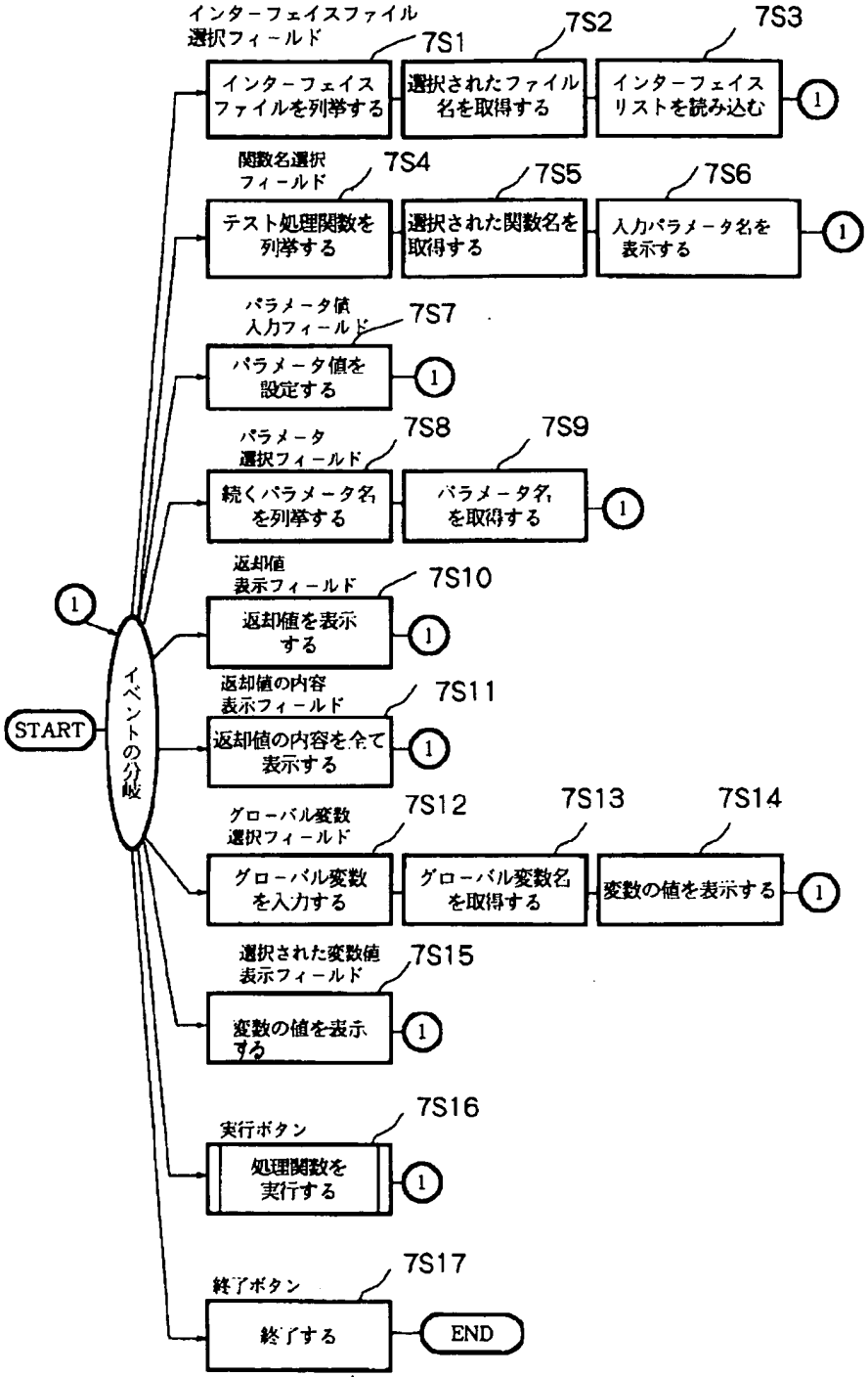
【図4】


START
初期インターフェイス ファイルの読み込み
4S1
指定されたソース ファイルの読み込み
4S2
インクルードファイル を読み込む
4S3
ソースファイル名を セクション名に追加する
4S4
インクルードファイル からtypedef宣言を取り出す
4S5
取り出したtypedef宣言型 を新規追加する
4S6
4S7
キーワードがあるか？
NO
YES
4S8
プロトタイプ宣言文を 取り出す
インターフェイス・リスト に追加する
4S9
4S10
ファイルポインタを先頭 に戻す
4S11
ファイルの終了か？
YES
NO
4S12
処理関数の開始文字を 検索する
4S13
関数が見つかったか？
NO
YES
4S14
プロトタイプ宣言文を 取り出す
4S15
インターフェイス・リスト に追加する
4S16
グローバル変数の アドレスを取得する
END

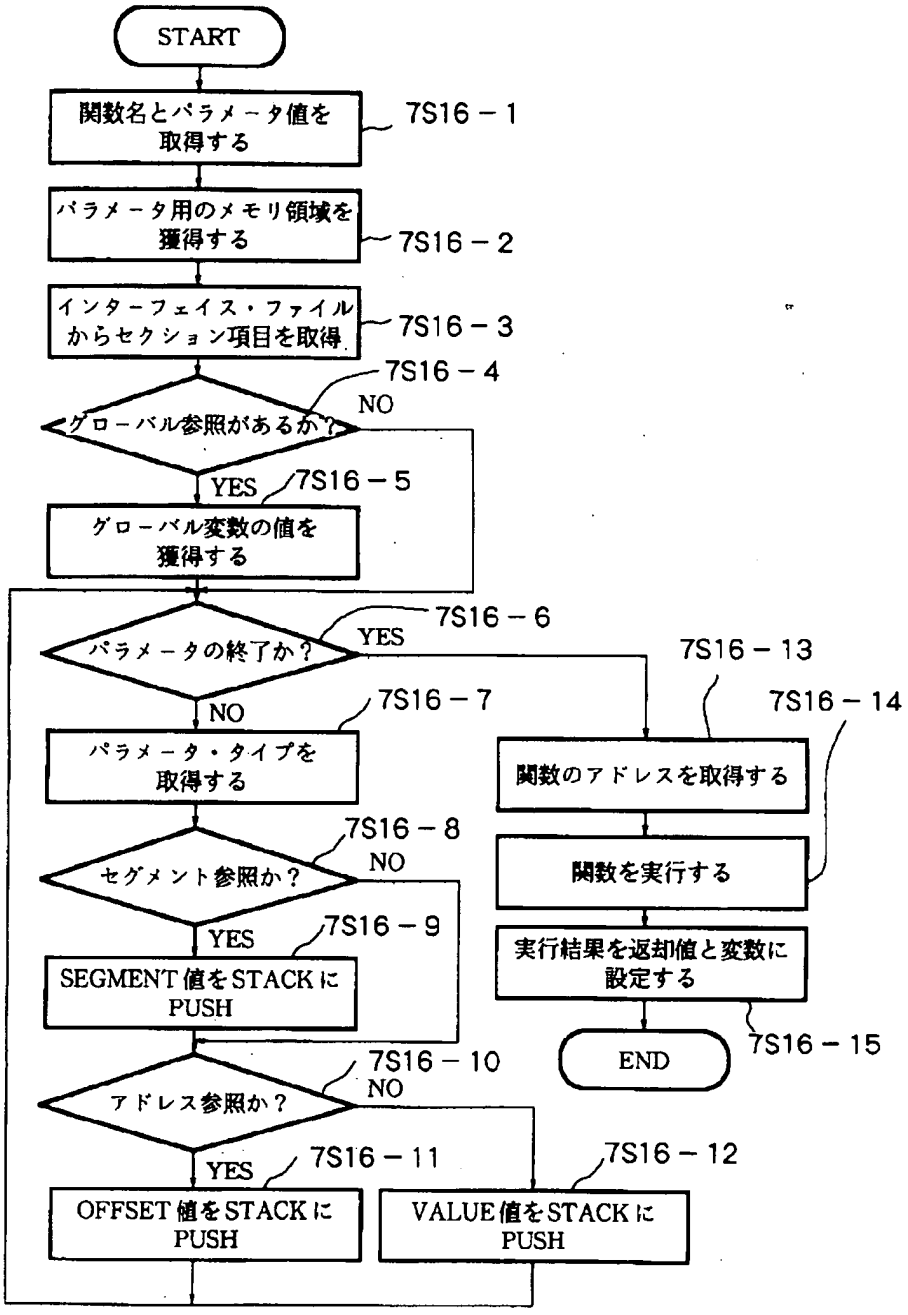
【図6】


START
イベントの分岐
モジュールファイル選択フィールド
6S1
スクロールボタンが押されたならば、ファイルを列挙する
6S2
選択されたファイル名を取得する
関数名選択フィールド
6S3
スクロールボタンが押されたならば、関数を列挙する
6S4
選択された関数名を取得する 前項目を保存する
6S5
インターフェイス・リストからセクション項目を取得する
6S6
各フィールドに値を設定する
変数名入力フィールド
6S7
変数名を取得する
変形型タイプ選択フィールド
6S8
スクロールボタンが押されたならば、型タイプを列挙する
6S9
選択された型タイプを取得する
パラメータ番号表示フィールド
6S10
パラメータの引渡しの順序を表示する
パラメータ追加フィールド
6S11
追加されるパラメータの引渡し順序を列挙する
6S12
選択された引渡し番号を取得する
グローバル変数入力フィールド
6S13
グローバル変数名を入力する
6S9
選択された型タイプを取得する
モジュールファイル読み込みボタン
6S14
選択されたモジュールファイルを読み込む
6S15
モジュールファイル名からインターフェイス・リストを得る
6S6
各フィールドに値を設定する
インターフェイスファイル保存ボタン
6S16
設定されたフィールドデータをインターフェイスファイルに保存する
END
1

【図7】


インターフェイスファイル
選択フィールド
7S1
7S2
7S3
インターフェイス
ファイルを列挙する
選択されたファイル
名を取得する
インターフェイス
リストを読み込む
関数名選択
フィールド
7S4
7S5
7S6
テスト処理関数を
列挙する
選択された関数名を
取得する
入力パラメータ名を
表示する
パラメータ値
入力フィールド
7S7
パラメータ値を
設定する
パラメータ
選択フィールド
7S8
7S9
続くパラメータ名
を列挙する
パラメータ名
を取得する
返却値
表示フィールド
7S10
返却値を表示
する
返却値の内容
表示フィールド
7S11
返却値の内容を全て
表示する
グローバル変数
選択フィールド
7S12
7S13
7S14
グローバル変数
を入力する
グローバル変数名
を取得する
変数の値を表示する
選択された変数値
表示フィールド
7S15
変数の値を表示
する
実行ボタン
7S16
処理関数を
実行する
終了ボタン
7S17
終了する
START
END
イベントの分岐
1

【図8】


START
関数名とパラメータ値を取得する
7S16－1
パラメータ用のメモリ領域を獲得する
7S16－2
インターフェイス・ファイルからセクション項目を取得
7S16－3
7S16－4
グローバル参照があるか？
NO
YES
7S16－5
グローバル変数の値を獲得する
7S16－6
パラメータの終了か？
YES
NO
7S16－7
パラメータ・タイプを取得する
7S16－8
セグメント参照か？
NO
YES
7S16－9
SEGMENT値をSTACKにPUSH
7S16－10
アドレス参照か？
NO
YES
7S16－11
OFFSET値をSTACKにPUSH
7S16－12
VALUE値をSTACKにPUSH
7S16－13
関数のアドレスを取得する
7S16－14
関数を実行する
実行結果を返却値と変数に設定する
END
7S16－15